



School of  
**informatics**



# System Administration and CDDLM

PAUL ANDERSON [dcspaul@inf.ed.ac.uk](mailto:dcspaul@inf.ed.ac.uk)  
EDMUND SMITH [esmith4@inf.ed.ac.uk](mailto:esmith4@inf.ed.ac.uk)

# Overview

- Motivation ...
  - Application configuration implies system configuration
  - They have very similar goals - can they learn from each other?
  - Is CDDL M applicable to fabric configuration ?
- What is system configuration ?
- Practical experiences with system configuration
  - Specifying autonomic behaviour
  - Federated configuration and conflict resolution
  - Evolving configurations
  - Managing configuration security
  - Complexity and usability
- System configuration projects
  - OGSAConfig & LCFG
- Conclusions

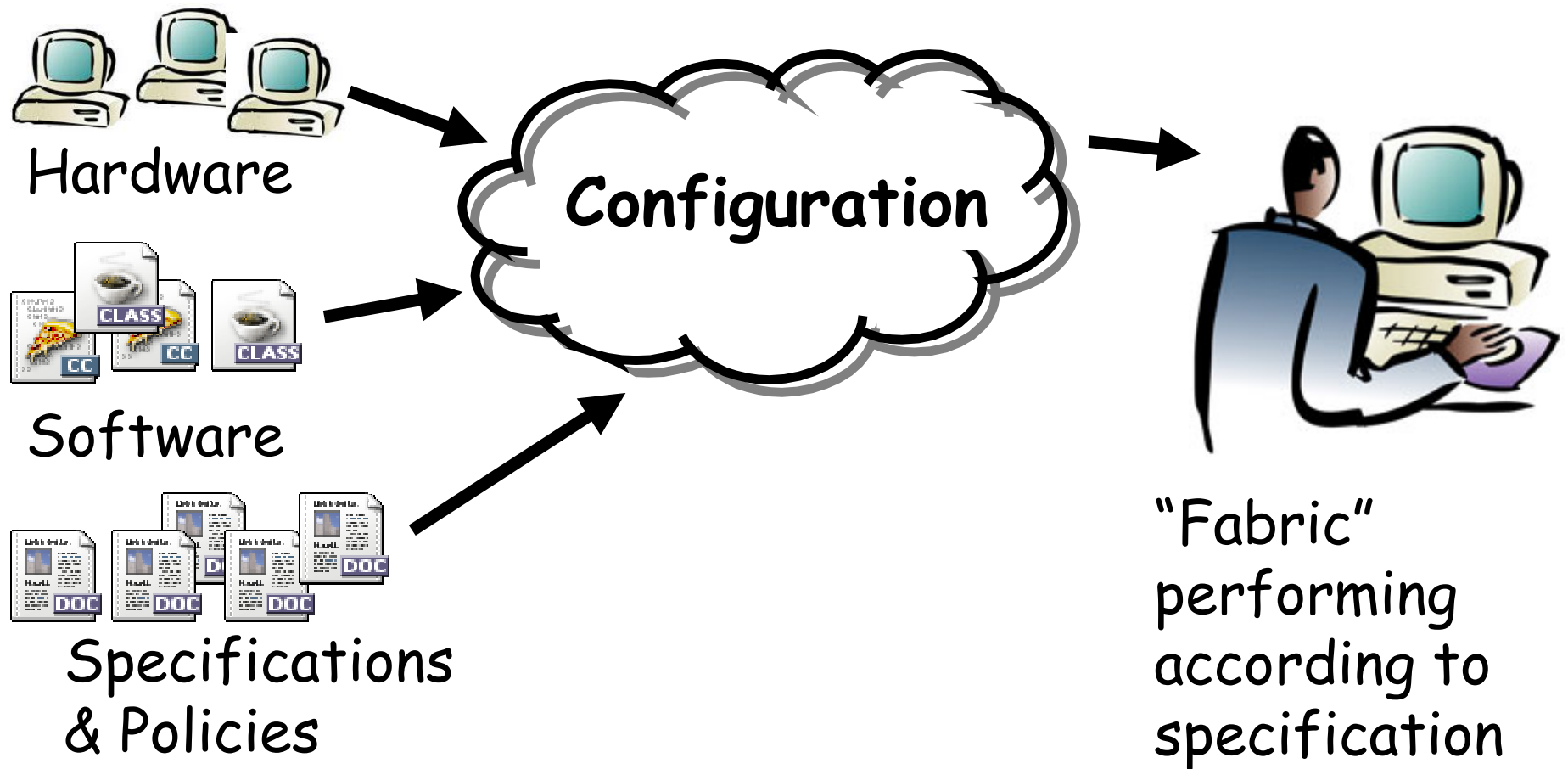
# Motivation ...

- System configuration and application configuration are dependent and must be integrated ...
- Applications depend heavily on fabric services
  - Authentication, names services (DNS,LDAP), firewalls
  - The ability to reconfigure these services in conjunction with the applications is necessary for overall fault tolerance
- Non-integrated configuration systems are a major source of failure
  - Configuration mismatch occurs when one is changed without the other

# Motivation

- System configuration and CDDLML goals are very similar
  - Configuration of distributed, federated services
  - Perhaps they can learn from each other ?
- Can system configuration benefit from CDDLML?
  - Development is currently inhibited by lack of standards
  - Fabric management is becoming more federated
  - Languages such as SmartFrog are more advanced than current languages
- Can CDDLML benefit from system configuration?
  - Fabric management has a lot of practical experience
    - Large, existing production systems
  - And a good understanding of important general configuration problems
    - Many of these are likely to be encountered in CDDLML
    - Some of these are solved, some are open research areas

# System Configuration



# System Configuration

- Starting with:
  - Several hundred new PCs with empty disks
  - A Repository of all the necessary software packages
  - A specification of the required service
- Load the software and configure the machines to provide the service
  - This involves many internal services - DNS, LDAP, DHCP, NFS, NIS, SMTP, Web ...
- Reconfigure the machines as the required service specification changes

# Fabric characteristics

- A typical installation might involve
  - 1000 nodes with 5000 parameters each
  - 50 configurable "components" per node
  - 10s of people involved in different aspects of the overall fabric configuration
- Many services are critical and long-lived
- Configurations evolve continuously
  - "Asymptotic configuration" is common
- Structuring and managing the configuration information in a "federated" environment is a major problem

# System configuration tools

- Current practice in configuration tools appears to favour the following properties:
- Declarative specification languages
  - Research is working to raise the level of these languages
- “Convergent” instantiation of the configuration without an explicit lifecycle
- Central authority for configuration data
  - Research is moving towards more distributed authority

# Practical configuration experiences



Declarative specifications for Autonomic systems

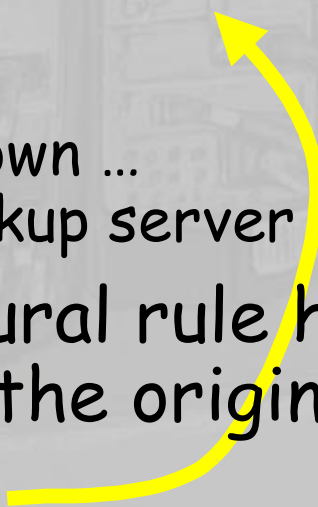
Security

Continuous reconfiguration

Federated configurations and conflict resolution

Complexity and usability

# Autonomics

- Traditional "fault-tolerance" systems for fabric management are usually based on event-action rules. For example:
    - A declarative configuration:
      - Hosts X, Y and Z are my web servers
    - An event-action rule:
      - If a web server goes down ...
      - Then configure the backup server S as a web server
    - Note that the procedural rule has broken the declarative nature of the original specification
      - This is no longer true
- 

# High-level specifications

- Some systems use the feedback to modify the original specification
  - This still does not provide a definitive declarative description
- We believe that facilities to describe requirements in a “looser” high-level way are necessary:
  - Hosts X, Y, Z, S are potential web servers
  - The fabric should contain 3 live web servers
- Fault-tolerance systems are free to make deployment decisions without changing the requirements specification.

# Conflict Resolution

- In any large system, different people will be responsible for different "aspects"
  - Individual configuration parameters are often affected by several different "aspects"
    - The DNS manager may want to define the available DNS servers
    - A host manager might want to specify that all services come from servers on the same network
    - A server manager may want to override this, or add some other constraint
- Current systems do not resolve conflict well:
  - Throw an exception?
  - Give implicit precedence to some aspect ?
    - Simple "default hierarchies" are insufficient
  - Allow arbitrary procedural code to decide?

# Aspect composition

- The language forces explicit values to be specified:
- Aspect A
  - Use server Y
- Aspect B
  - Use server X
- This conflict is irreconcilable without human intervention because we don't know the intention
- The user really only wants to say ...
- Aspect A
  - Use any server on my Ethernet segment
- Aspect B
  - Use one of the servers X,Y or Z
- These constraints can be satisfied to
  - Use server Y (assuming Y is on the right segment)

# Evolving configurations

- Most modern fabric configuration systems are based on “convergent” configuration:
  - The configuration is specified in a declarative way
  - The tool continuously adjusts the real configuration to match the specification
    - Either the specification, or the “reality” might change
    - (hardware fails, software changes)
  - There is no explicit “lifecycle”
- Service are long-lived and reconfigure continuously
  - Configuration is frequently “asymptotic”
- Sequencing of related changes must be handled explicitly if it is important

# Configuration security

- Most fabrics have good internal authentication infrastructures
- However ...
  - Fine-grained authorisation control is needed
  - There are problems integrating "machine-generated" configuration information
    - "Who is the author of this fragment of configuration?"
  - Tracking and controlling security dependencies is very hard when changing parameters which affect low-level fabric services
    - "What is the consequences of changing this parameter?"

# Complexity and usability

- Configuration tools are intended to enable deployment of reliable and correct configurations
  - Configuration errors are a major source of failures
  - We must consider human factors
- We have strong evidence that (apparent)? complexity is inhibiting the uptake of fabric configuration tools
  - Configuration errors with existing tools are most often due to a lack of clarity in the semantics
- We have some concerns about the complexity of the CDDL M proposals

# Trust and explanation

- A successful tool needs considerable freedom to make configuration decisions
- Since these are likely to have a significant impact on the fabric, administrators are very nervous about delegating the necessary authority
- Tools need to be trustworthy and good interfaces need to be provided which present views of the system tailored to different classes of user
- Tools must also be able to provide useful explanations for their actions
  - Why is the divisional web server running from my laptop?

# Fabric configuration projects



## **OGSAConfig**

Dynamic fabric configuration in response to application requirements

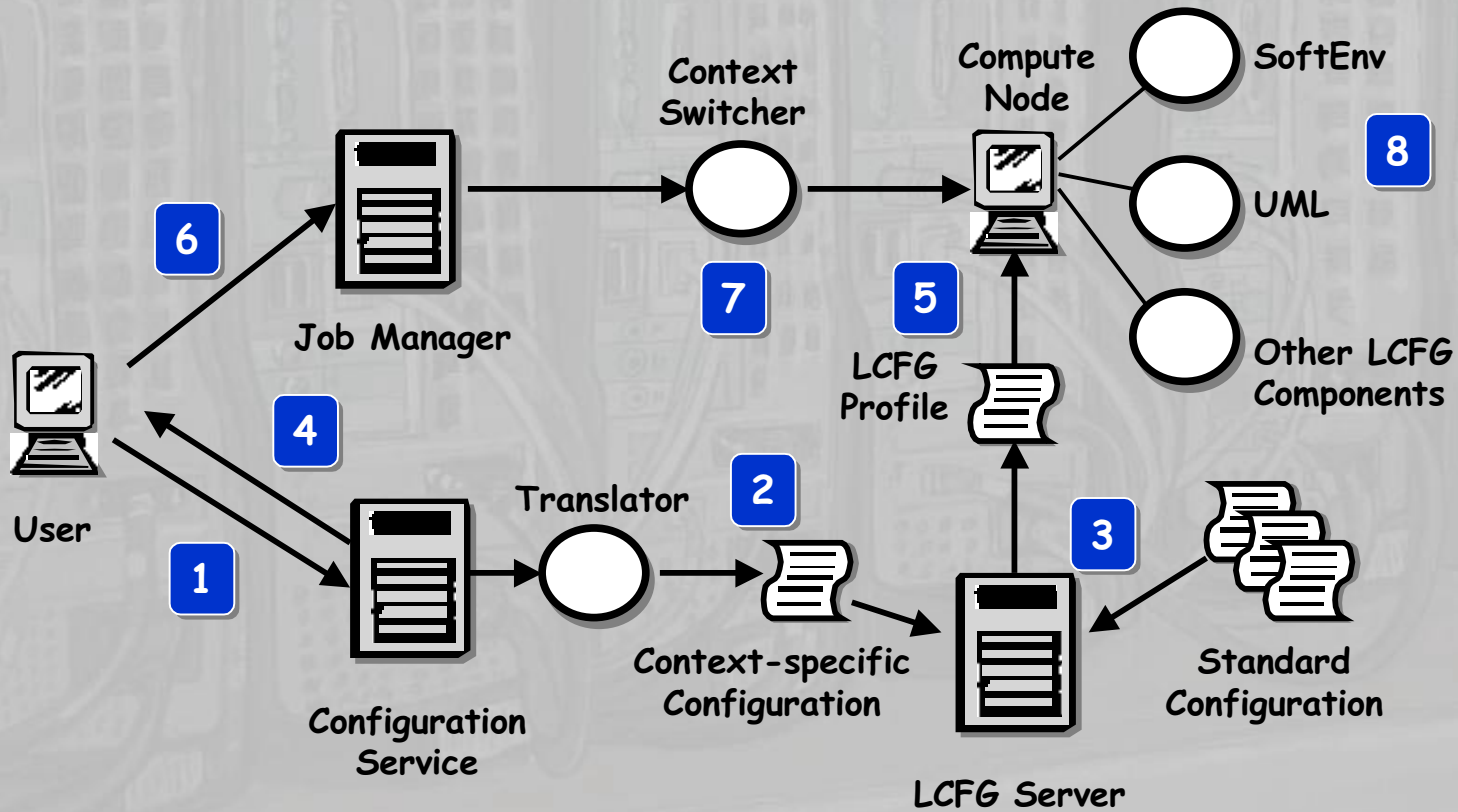
## **LCFG**

A fabric configuration tool

# OGSAConfig

- Dynamic reconfiguration of Grid fabric nodes to meet application requirements
  - Eg, package versions, kernel parameters, services
- Applicable to existing e-Science grids
  - Configuration requirements negotiated with a configuration broker
  - Application deployment under existing job manager control
- Investigated use of CDDLML
  - Not a good match for this application due to emphasis on host-oriented deployment

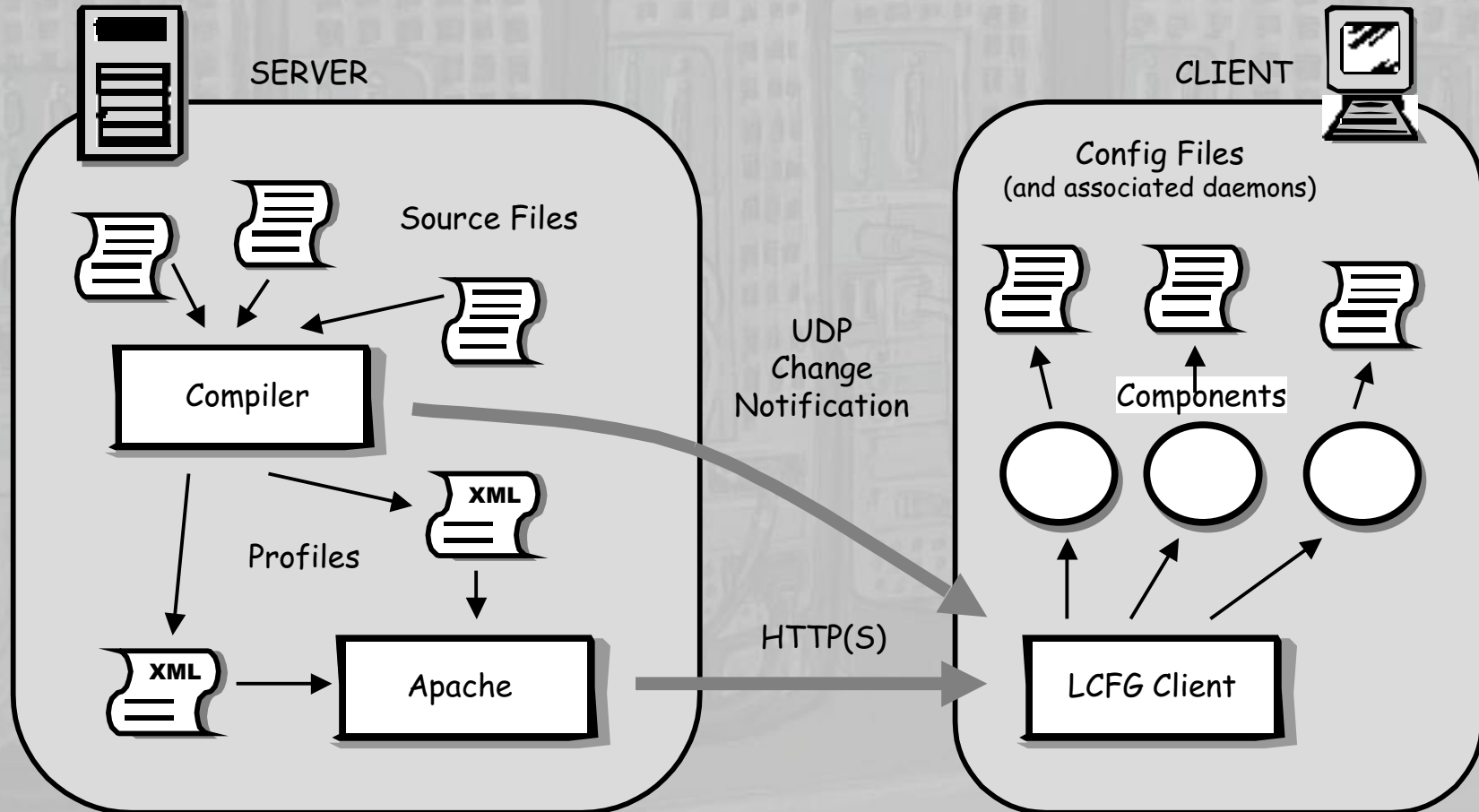
# OGSAConfig Architecture



# LCFG

- A system configuration tool developed about 10 years ago as a production fabric management system for Computer Science at Edinburgh University
- The original paper established some important principles:
  - [http://www.lcfg.org/doc/LISA8\\_Paper.pdf](http://www.lcfg.org/doc/LISA8_Paper.pdf)
- An ongoing testbed for configuration research
  - Only recently outlived its usefulness for this purpose
- Available as open source and used at several other sites

# LCFG Architecture



# Some LCFG characteristics

- LCFG mandates a single source of configuration information
  - All external-generated information must be routed through the central server
  - We consider this a problem
- The central compilation provides the ability to construct individual node configurations from higher-level descriptions of "services"
  - Automatically create firewall holes when adding a web server
- LCFG is "convergent" - there is no deployment "lifecycle"

# Conclusions

- There are many similarities between CDDL M aims and those of existing system configuration tools
  - It will be beneficial to maintain an exchange of knowledge, and to adopt common technologies and approaches where possible
- It is important for application configuration to inter-operate with fabric configuration
  - Existing fabric management technologies are not a good match for CDDL M
  - It may be worth considering how this interoperation may be achieved
- Some important configuration problems are not yet well-understood
  - There may be a danger of CDDL M attempting to standardise solutions to these too soon

# Links

- Paul Anderson homepage
  - Online copy of these slides, related paper and other research
  - <http://homepages.informatics.ed.ac.uk/dcspaul/>
- The OGSAConfig Project
  - <http://groups.inf.ed.ac.uk/ogsaconfig>
- The LCFG Project
  - <http://www.lcfg.org>
- LssConf
  - System configuration workshops and mailing
  - <http://homepages.inf.ed.ac.uk/group/lssconf>
- GridWeaver
  - Autonomic reconfiguration using LCFG and SmartFrog
  - <http://www.gridweaver.org>



School of  
**informatics**



# System Administration and CDDLM

PAUL ANDERSON [dcspaul@inf.ed.ac.uk](mailto:dcspaul@inf.ed.ac.uk)  
EDMUND SMITH [esmith4@inf.ed.ac.uk](mailto:esmith4@inf.ed.ac.uk)